НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль

(підпис) (ініціали, прізвище)

“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 121 Інженерія програмного забезпечення

на тему «Система аналізу та формування звітів госпіталізованих хворих»

Виконав (-ла): студент (-ка) 4 курсу, групи ТВз-61

Паньківський Олексій Вікторович _____

(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к. т. н., Крячок Олександр Степанович _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доцент к. т. н., Реуцький Микола Олександрович _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень (бакалаврський)

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студента
Паньківського Олексія Вікторовича
(прізвище, ім’я, по батькові)

1. Тема роботи «Система аналізу та формування звітів госпіталізованих хворих» керівник роботи Крячок Олександр Степанович, к. т. н., доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від «25» травня 2020р. №1168-с

2. Строк подання студентом роботи «10» червня 2020 р.

3. Об’єктом дослідження є процес аналізу та формування звітів госпіталізованих хворих медичного закладу.

4. Предмет дослідження – автоматизація процесу аналізу та формування звітів госпіталізованих хворих медичного закладу.

5.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз аналогічних систем в системі охорони здоров’я, засоби розробки Hospital Manager Helper, реалізація програмної системи, інструкція користувача програмного забезпечення

6. Перелік ілюстративного матеріалу діаграма моделі; діаграма класів; діаграма прецедентів; результати виконання програми

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання «11» жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	26.01.2020 р.	
2.	Вивчення та аналіз задачі	01.03.2020 р. – 20.03.2020 р.	
3.	Розробка архітектури та загальної структури системи	20.03.2020 р. – 25.04.2020 р.	
4.	Розробка структур окремих підсистем	25.04.2020 р. – 10.05.2020 р.	
5.	Програмна реалізація системи	10.05.2020 р. – 25.05.2020 р.	
6.	Оформлення пояснювальної записки	25.05.2020 р. – 10.06.2020 р.	
7.	Захист програмного продукту	10.06.2020 р.	
8.	Передзахист	10.06.2020 р.	
9.	Захист	17.06.2020 р.	

Студент _____ Паньківський О. В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Крячок О. С.
(підпис) (прізвище та ініціали,)

Анотація

Метою дипломного проекту є створення програмного забезпечення, що є аналогом медичної інформаційної системи, яка б відповідала вимогам лікаря – спеціаліста.

Створена система, яка компенсує основні недоліки існуючих медичних інформаційних систем. Також наведено стислий огляд засобів, які використовувались для створення програмного забезпечення, описано головні переваги обраних методів для рішення поставленої задачі.

Система призначена для використання лікарем, обліку та аналізу інформації про пацієнта. Інформація зберігається у базі даних, з якої за необхідним запитом можна отримати повну інформацію про пацієнта. Дипломний проект виконано на 51 аркушах, він містить перелік посилань на використані джерела з 31 найменувань. У роботі наведено 16 рисунків.

Ключові слова: МІС, лікар, система, медичний заклад, документообіг, Java, база даних.

Abstract

The purpose of the diploma project is to create software that is analogous to medical information a system that would meet the requirements of a specialist doctor.

A system has been created that compensates for the main shortcomings of existing medical information systems. There is also a brief overview of the tools used to create software, describes the main advantages of the selected methods for solving the problem.

The system is designed for use by the doctor, accounting and analysis of patient information. The information is stored in a database from which, if necessary, complete information about the patient can be obtained. The diploma project is executed on 51 sheets, it contains the list of references to the used sources from 31 names. There are 16 pictures in the work.

Key words: MIS, doctor, system, medical institution, document circulation, Java, database.

Зміст

Перелік скорочень, умовних позначень, термінів

МІС – медична інформаційна система.

ЦБД – Центральна база даних(ДП «Електронне здоров’я»).

API – application programming interface або програмний інтерфейс додатку.

IDE - Integrated Development Environment або інтегроване середовище розробки.

Heap або купа – використовується JRE для виділення пам'яті під об'єкти і класи.

JRE – Java Runtime Environment або середовище виконання для Java, мінімальна. реалізація віртуальної машини, необхідних для виконання java додатків.

Swing – бібліотека для розробки графічних інтерфейсів мовою Java.

МІС – Медична інформаційна система.

ПЗ – програмне забезпечення.

ПК – персональний комп'ютер.

СЕД – система електронного документообігу.

СУБД – система управління базами даних.

Вступ

Завжди легше попереджати загрозу чим ломати голову над вирішенням проблеми. Превентивні заходи мають той самий ефект в системі охорони здоров'я. Витрати на охорону здоров'я зростають з кожним роком. Це також пов'язано з тим що населення більшості розвинених країн старішає – народжується менше дітей та збільшується кількість людей які долають поріг похилого віку.

З розвитком інформаційних технологій комп'ютери та програмне забезпечення дозволили зробити стрибок у напрямку зростання продуктивності праці. Зараз практично не має сфер, яких не торкнувся комп'ютер з іншим автоматизованим обладнанням. Однак система охорони здоров'я в Україні лишається недостатньо автоматизованою, аби надавати якісніші послуги.

Паперові звіти та інший облік про стан пацієнта, його процедури, аналізи, система діагностики – все це може та має бути автоматизовано. Тоді лікарі будуть мати більше часу на порятунок та лікування пацієнтів. Міцна система охорони здоров'я є запорукою підвищення справедливості та якості життя. Не дарма ж в усілякі індекси включають якість надання медичної допомоги. Можна сказати, що сильна та дієва охорона здоров'я є запорукою успіху для будь – якої країни, Україна не виняток.

Об'єктом дослідження в цьому проекті є процес обробки діагностичної інформації госпіталізованих хворих.

Предметом дослідження є автоматизація процесу наведеного вище.

Мета роботи полягає в створенні автоматизованої системи яка дозволяє аналізувати та певним чином формувати звіти госпіталізованих хворих.

Для виконання поставленого завдання необхідним є:

- дослідити процес збору та обробки інформації;
- виявити медичні системи які могли б автоматизувати процес;
- визначити можливість та доцільність автоматизації процесів;

- обрати методи та засоби для побудови проекту;
- уточнити набір функцій, які має виконувати проект;
- визначити вимоги до програмного та апаратного забезпечення;
- розробити графічний інтерфейс.

Створення загальної та всеохоплюючої інформаційної системи є необхідною умовою проведення медичної реформи.

1) Дослідження ринку, перевірка ідеї проекту

1.1) Опис проблеми загалом

Діагностичні дані пацієнтів збільшуються, кількість МІС зростає. Як результат, кількість інформації про пацієнта збереженої на цифрових носіях – збільшується. Потреба у функціональному змісті цих систем також зростає. Незважаючи на те, що системи ІМС зараз широко застосовуються в Україні, їх область застосування наразі включає лише первинну медицину. Значна частина лікарів досі використовують паперові носії у своїй роботі. У 2018 році в Україні буде впроваджена електронна система охорони здоров'я eHealth, яка гарантує права пацієнтів щодо якості медичних послуг та оптимізує взаємодію між пацієнтом та лікарем шляхом автоматизації, медичного обліку та електронного управління медичною інформацією.

EHealth складається з центральної бази та великого розмаїття МІС. Центральна база даних Україні забезпечує прозорість витрат охорони здоров'я та можливість руху інформації без паперів з поступовим переходом до електронного обліку, включаючи електронні рецепти, електронні картки та електронні довідки, створювати нові електронні послуги, створення ділового середовища, створення інноваційних продуктів у медицині та просування медичного ринку ІТ загалом.

Завдяки швидкому доступу до всієї інформації про пацієнта, лікар отримує цілісне уявлення про ваше здоров'я, а повний анамнез в електронній формі допомагає поставити правильний діагноз. Більшість медичних послуг можна отримати, не виходячи з дому. Модель фінансування медичних закладів з допомогою системи eHealth кардинально змінилася. Діє принцип – «гроші йдуть за пацієнтом». Система дозволяє контролювати ефективність, з якою витрачаються державні кошти. Перш за все, eHealth охопить основну допомогу: лікарі загальної практики, терапевти та педіатри. Пацієнти підписують заяви від вибраних лікарів, а

лікарі реєструють їх у системі. Це допоможе державі оплатити роботу лікаря з кожним пацієнтом. У 2019 році Україна перебуває на фінішній прямій впровадження першої частини електронної системи охорони здоров’я, включаючи завершення реєстрації медичних установ та майже підписання заяв лікарів та пацієнтів. Планується відновити репутацію медичного персоналу на додаток до збільшення заробітної платні, завдяки:

- оновлення парку машин;
- проведення додаткових тендерів для закупівлі інфраструктурного обладнання;
- автоматизація диспетчерських служб.

Як наслідок медичної реформи Україна повинна охоплюватись розгалуженою мережею сучасних центрів контролю. Це означає, що на дзвінки відповідають швидше, а служби швидкої допомоги приїздять вчасно.

Існують особливості, які сповільнюють впровадження системи, зокрема відсутність унікального ідентифікатора для пацієнтів, неповна взаємодія між національними регістрами, неповна кількість реєстрів та відсутність фахівців з автоматизації та управління змінами. Наразі використовуються та впроваджуються лише певні елементи eHealth. Досі не існує розробленої та затвердженої концепції та національної стратегії щодо електронної системи охорони здоров'я, архітектури та мандату, яка була чітко затверджена на правовому рівні. Не існує міжнародних узгоджених та затверджених числових стандартів щодо зберігання, обробки та передачі медичних даних, класифікаторів та каталогів.

Слід зазначити, що важливо захищати особисті дані пацієнтів. Вся інформація або спектр інформації про пацієнта, яка занесена до декларації про вибір лікаря, є особистими даними пацієнта (ім'я, дата народження, реєстраційний номер податкового рахунку, номер та порядковий номер). Закон України "Про захист персональних даних" від 1 червня 2010 р. № 2297-VI [1] (далі: закон) регулює правовідносини щодо захисту та обробки персональних даних і спрямований на захист основних прав та свобод.

Відповідно до закону, персональні дані - це інформація або сукупність інформації про особу, яку можна конкретно ідентифікувати. Коли електронна лікарня та електронна медична карта починають працювати в системі, персональні дані обробляються для забезпечення процесу лікування та покращення роботи електронної системи охорони здоров’я. Підписуючи декларацію з лікарем, людина погоджується, що їхні персональні дані є доступними в електронній системі тим лікарям, до яких вони звертатимуться за медичною допомогою. Персональні дані пацієнтів повинні підпадати під дію Закону про медичну конфіденційність та забезпечувати захист цих персональних даних. Медичні працівники зобов’язані жодним чином не розголошувати довірені їм або відомі у зв'язку з виконанням професійних чи службових чи трудових зобов'язань особисті дані, якщо це не передбачено законом. Доступ до даних про пацієнта, що містяться в електронній системі охорони здоров’я, можливий лише за письмовою згодою пацієнта (його законного представника) або на бланку, що дозволяє зробити висновок за згодою. Без згоди доступ до інформації про пацієнтів можливий лише у таких випадках:

- загроза життю пацієнта;
- згідно рішення суду.

1.2) Технологія електронного документообігу

З початку бурхливого впровадження технологій у світі головну роль відіграли принципи створення, зберігання та передачі даних. Електронні системи управління документами можуть заощадити значний час порівняно з паперовими документами та дати можливість автоматизованої обробки даних. У галузі медичної допомоги ми можемо побачити залежність того, що одужання пацієнта безпосередньо залежить від своєчасного і правильного діагнозу. Слід зазначити, що лікар приділяє значно більше уваги пацієнтам за наявності електронних систем управління документами, оскільки лікар вже не витрачає багато часу на роботу з документами, а швидше дбає про стан своїх пацієнтів. Електронні системи

управління підвищують ефективність. Закон сьогодні встановлює загальні вимоги до інформації та документів в електронній системі охорони здоров’я. Створення, введення, відображення інформації та документів у центральній базі даних, а також внесення змін та доповнень здійснюються користувачами відповідно до прав доступу. Інформація та документи створюються та вносяться до центральної бази даних українською мовою. Рух документів в електронній системі охорони здоров’я здійснюється відповідно до вимог законодавства України «Про електронні документи та електронний документообіг». Всі електронні документи, внесені до центральної бази даних, повинні супроводжуватися електронним підписом автора або підписом, який відповідає власноручному підпису відповідно до закону. Інформація під час обробки та обміну в електронній системі охорони здоров'я повинна залишатися цілісною. Це забезпечується захистом від несанкціонованих дій, які можуть призвести до випадкової чи навмисної зміни чи знищення, зокрема шляхом накладення на електронний підпис автора або підпису, який відповідає власноручному підпису. відповідно до закону. Державні класифікатори, номенклатури та затверджені каталоги використовуються для введення інформації та документів у центральну базу даних у встановленому законодавством порядку, зокрема класифікаціях та спеціальних списках. Адміністратор надає технічну підтримку для використання таких класифікацій, номенклатур, каталогів та списків. Кожен документ та інформація, внесені до центральної бази даних, автоматично реєструються в єдиному реєстрі. Ідентифікатор особи - це унікальний реєстраційний номер у демографічному реєстрі держави [2]. Слід зазначити, що переваги використання електронного управління документами є вагомими. Сильними аргументами для електронного управління документами є:

- швидкий пошук медичних даних;
- протидія халатності персоналу, що може призвести до втрати медичних даних;
- шифрування підвищує безпеку;
- зберігання інформації в базі.

Однак системи електронного документообігу мають свої недоліки. Основними недоліками є:

- значні початкові витрати на впровадження СЕД;
- термін запуску СЕД;
- необхідність підвищення кваліфікації персоналу для роботи з СЕД;
- можливість втрати інформації.

Вцілому будь – яка сфера, яка отримує доступ до системи електронного документообігу має значний стрибок у продуктивності праці. Це перехід на якісно інший рівень. Так, впровадження системи дороговартісне, але роблячи крок до впровадження цієї технології , установа швидко окупить її з часом.

1.3) Аналіз існуючих систем

Гіганти цифрової індустрії розуміючи невідворотність проникання інформаційних технологій в заклади охорони здоров'я, вирішили прийняти участь у боротьбі за частину ринку. На цей час впровадженими є Google Health, Microsoft Health Vault та Dossia (організовані Intel, Wal-Mart та AT&T). Співпраця цих важковаговиків дозволить об'єднати усю медичну інформацію та створити єдиний стандарт роботи з діагностичними даними пацієнтів. В подальшому це об'єднання при значних дослідженнях та витратах ресурсів дасть можливість запустити систему штучного інтелекту зі величезною базою знань. Використання цієї технології важко уявити, вона виходить за межі охорони здоров'я як такої.

Медична мережа дозволяє інтегрувати дані з різних систем в особисту картку пацієнта. Компанії поставили перед собою завдання створити інтегроване мережеве – середовище, в якому зберігається первинна діагностична інформація пацієнта.

Стандарт HL7 підтримується Google Health та Microsoft Health Vault. Це дозволяє вести обмін інформацією між базами різних систем.

На ринку України є державні та приватні медичні центри. Кожен медичний центр підтримує власну базу відвідувань кожного пацієнта, особливо це стосується приватних закладів. Разом з тим пацієнти продовжують мати власні паперові картки з медичною документацією. Звести всю інформацію до єдиної бази даних або хмари здається неможливим. Кожна процедура обміну медичними даними змушує гаяти час, що зменшує шанси пацієнта на виживання під час надання швидкої медичної допомоги. Оптимізація та об'єднання всіх історій хвороб пацієнта, є важливим в першу чергу для держави. Здорова нація – успішна країна. Усі медичні центри, що працюють в Україні, повинні мати доступ до зведеної бази даних. Наступним завданням, з яким потрібно зіткнутися, є доступ до Інтернету з розумною швидкістю всіх медичних установ. Зрештою, обмін діагностичними даними у великих кількостях при низькій швидкості зв'язку або великому навантаженні мережі може призвести до значної трати часу.

Ще одним вирішенням проблеми може стати впровадження стандартів зберігання на основі CDA (Clinical Document Architecture). Це дозволить мати в базі тільки найважливішу інформацію та посилання на медичні картки в інших центрах охорони здоров'я. Ця модем менше залежить від каналу зв'язку.

Хоча система MIS працює не у всіх сферах медичних установ України перед лікарями стоїть складне завдання заповнити десятки форм. Переважна більшість медичних інформаційних систем призначені для лікарів загальної практики, хоча також існує великий потік медичних даних від лікарів-спеціалістів, який потрібно збирати та обробляти. На сьогодні існує кілька десятків різних систем, найбільш відомою серед них є "Helsi" [5] (рисунок 1).

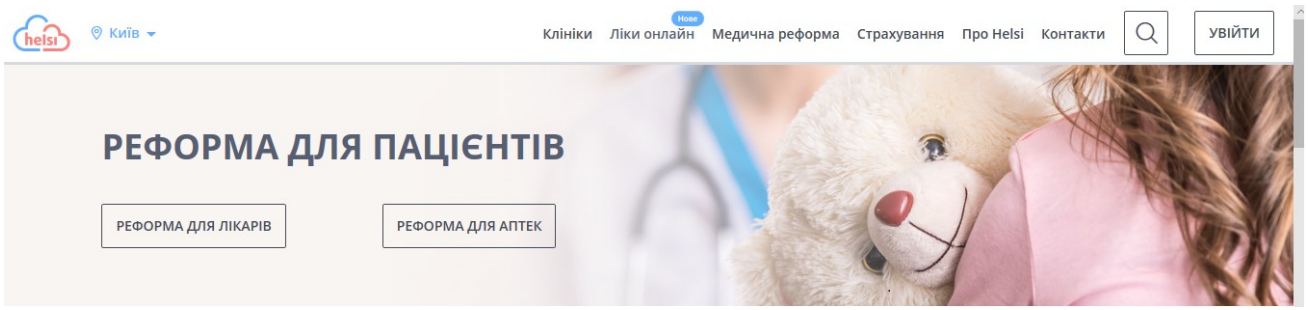


Рисунок 1 – Helsi.me

Ця система дуже відома оскільки переважна більшість населення користується державними центрами охорони здоров'я. Доступ до системи пацієнт отримує після підписання декларації з персональним лікарем.

Система має простий інтуїтивний інтерфейс. У випадку якщо клієнт забув пароль, відновити доступ можна за допомогою одноразового паролю через смс. Маю відзначити система іноді самостійно надсилає одноразовий пароль під час входу, як елемент додаткового захисту. Недоліком цього є те, що пацієнт матиме труднощі з входом до системи, якщо загубить телефон.

Система працює з електронним рецептом та створено напрямок страхової медицини. Страхова медицина дозволить залучити додаткові кошти та побороти проблеми з недофінансуванням в галузі охорони здоров'я. За рахунок цього кошти матимуть цільове призначення та відповідно будуть рости і розвиватися ті сервіси та послуги якими користуватимуться пацієнти. Державі лишиться фінансування стратегічних та перспективних напрямків розвитку галузі за потреби.

Кожен пацієнт має власний кабінет (рисунок 2) де він може:

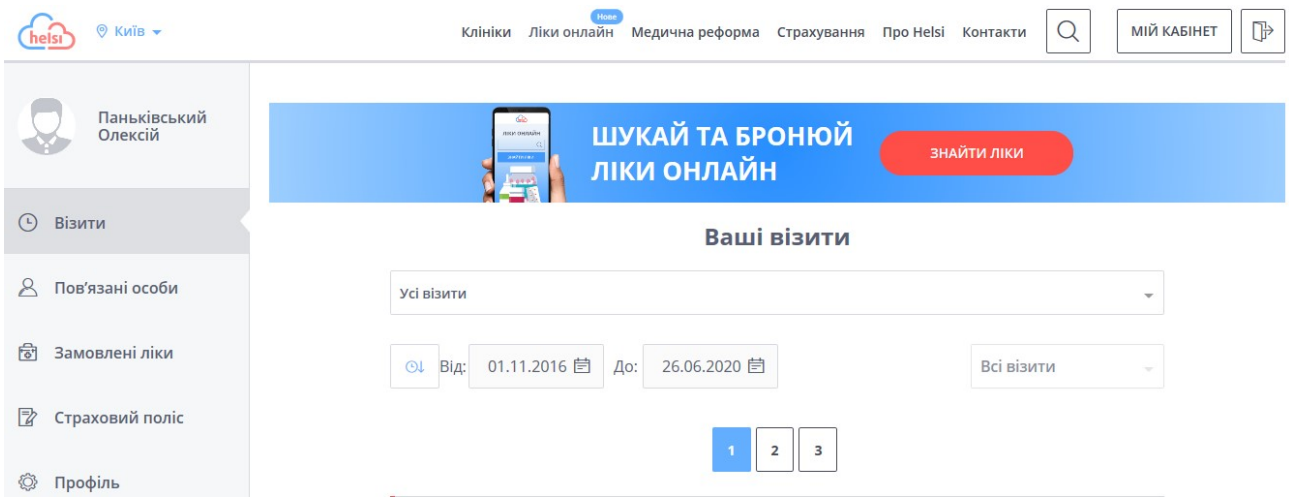


Рисунок 2 – Власний кабінет Helsi

- редагувати власний профіль;
- переглянути історію візитів за весь час;
- працювати з пов'язаними особами(наприклад : сім'я);
- замовляти ліки ;

- користуватися страховим полісом.

Єдиним на мою думку суб'єктивним недоліком Helsi є обмеженість співпраці зі страховими компаніями. Свій страховий поліс я не зміг долучити, аби протестувати таку можливість. В перспективі я впевнений це питання буде закрите.

1.4) Вимоги до проекту

Під час зустрічі з замовником було поставлене наступне технічне завдання:

Вимоги до системи вцілому:

- створення інтуїтивно зрозумілого графічного дизайну;
- авторизація та політика безпеки;
- заповнення зовнішніх форм статистики та збереження діагностичних даних пацієнтів для подальшої обробки;
- тестування та обробка помилок;
- виконання робіт на стороні замовника перед першим запуском;

Технічні та апаратні вимоги:

- ОС Windows 7,8,10;
- вимоги до оперативної пам'яті 2 гб або більше;
- вільне місце на жорсткому диску з базою не менше 200 гб;
- встановлена JVM;
- підготовлена до роботи база даних MS SQL;

Функціональність:

- **Навігація:**
  - внесення даних до бази та заповнення статистичних звітів;
  - пошук у базі даних та аналіз діагностичних даних певним чином;

- вихід з програми;

- **Внесення даних до бази та заповнення статистичних звітів:**

- додавання до бази даних звернень пацієнтів про кожний конкретний такий випадок;

- автоматизація заповнення та надсилання статистики через сайт;

- **Пошук:**

- створення запиту для аналізу інформації в базі даних;

- виведення результату запиту в зручній формі;

- збереження частини результатів роботи користувача за потреби.

Головне призначення запропонованої системи полягає в створенні буферної бази, що допоможе уникнути помилок при заповненні статистичних форм. Це також дозволить зберігати частину результатів і не починати все з початку. Лікарі та молодший медичний персонал зможуть більше часу приділяти пацієнтам а не статистиці. Також враховано факт, що успіх ІТ – проектів в галузі охорони здоров'я сильно залежить від сприйняття користувачем зручності інтерфейсу програмної системи[3], оскільки не всі користувачі маю однаковий досвід роботи з програмними системами.

Висновки до розділу

На ринку багато медичних систем, кожен зі своїми характеристиками. Велика кількість лікарів використовує такі системи у своїй роботі, що значно підвищує ефективність їх роботи та дає змогу приділяти значно більше уваги пацієнту. Для того, щоб кожен медичний заклад знаходив свою інформаційну систему серед існуючих, необхідно розвивати ринок IT - продуктів. «Helsi» привернула мою увагу серед аналізованих МІС. Я також повинен сказати, що я вже користувався цим і що я був задоволений. Виходячи з наведеного вище аналізу, ця система є ідеальним вибором для лікарів загальної практики. Однак є ще багато лікарів, які не можуть використовувати МІС під час роботи з наступних причин:

- відсутність інфраструктури (ПК, доступ до Інтернету);
- труднощі у налаштуванні існуючих МІС.

Окремо хочу відзначити запуск страхової медицини. Але цей напрямок потребує вдосконалення.

На цьому тлі необхідно вдосконалити діючу МІС або створити нове програмне забезпечення, яке виконує функції МІС у галузях медицини, для яких сучасні системи зараз не підходять.

2) Розробка Hospital Manager Helper

2.1) Інтерфейс користувача

Якщо ми розробляємо та розробляємо цифрові продукти таким чином, щоб люди, які ними користуються, могли легко досягти своїх цілей, вони будуть задоволені, ефективні та щасливі. Вони із задоволенням заплатять за нашу продукцію - і рекомендують іншим робити те саме. Якщо припустити, що ми можемо це зробити рентабельно, це перетвориться на успіх у бізнесі.

На перший погляд, ця ідея здається очевидною: зробіть людей щасливими, і ваша продукція матиме успіх. Чому ж тоді стільки цифрових продуктів настільки важкі та неприємні у використанні? Чому ми не всі щасливі та успішні, коли використовуємо їх? Чому, незважаючи на постійний марш швидших, дешевших і доступніших технологій, ми все ще так часто буваємо розчаровані?

Коротше кажучи, відповідь - відсутність дизайну як основної та рівноправної частини процесу планування та розробки продукції. Дизайн – це усвідомлене та інтуїтивне зусилля нав'язати змістовний порядок. Ми пропонуємо дещо детальніше визначення проектно-орієнтованої діяльності, орієнтованої на людину:

- Розуміння бажань, потреб, мотивацій та контекстів людей, які використовують продукти.
- Розуміння ділових та технічних можливостей, вимог та обмежень
- Використання цих знань як основи для планування створення продуктів, форма, зміст та поведінка яких корисні, корисні та бажані, а також економічно вигідні та технічно здійснені.

Цифрові продукти регулярно припускають, що люди грамотні з технологіями. Наприклад, у Microsoft Word, якщо користувач хоче перейменувати

документ, який він редагує, він повинен знати, що він повинен або закрити документ, або скористатися командою меню "Зберегти як ..." (і не забудьте видалити файл зі старим іменем). Ці поведінки несумісні з тим, як звичайний користувач думає про перейменування чогось; швидше, вони вимагають, щоб людина змінила своє мислення, аби більше нагадувати спосіб роботи комп'ютера.

Що стосується цифрових продуктів, планування включає розуміння того, як люди, які використовують продукт, живуть і працюють, а також розробляти поведінку та форму виробів, які підтримують та полегшують поведінку людини.

Свідоме включення дизайну передбачило піднесення сучасної тріади щодо розвитку продукції, визначеної Ларрі Кілі з групи "Доблін": спроможність, життєздатність та бажаність (див. Рис. 1-3). Якщо будь-який із цих трьох основ слабкий, продукт навряд чи витримає випробування часом.

Дизайн поведінки - це різного роду проблема, яка потребує більшого знання контексту, а не лише правил візуальної композиції та бренду. Це вимагає розуміння стосунків користувача з продуктом від придбання до кінця терміну експлуатації. Найважливішим є розуміння того, як користувач хоче користуватися продуктом

2.2) Проектування моделі програми

IDEF0 (Integration Definition for Function Modeling) – нотація опису бізнес-процесів. Заснована на методології SADT.

SADT (Structured Analysis and Design Technique, технологія структурного аналізу і проектування) - графічні позначення і підхід до опису систем. Розробка SADT почалася в 1969 році і була випробувана на практиці в компаніях різних галузей (аерокосмічна галузь, телефонія і т.д.). Публічно з'явилася на ринку в 1975 р і отримала дуже широке поширення в світі.

IDEF0 є результатом програми комп'ютеризації промисловості, яка була запропонована ВВС США. Автоматизація діяльності підприємств зажадала відповідних методик та інструментів. Перед тим, як розробляти програмне забезпечення, необхідно було чітко і зрозуміло описати бізнес-процеси (не можна автоматизувати хаос). Нотація може бути використана для моделювання широкого кола автоматизованих і неавтоматизованих систем.

Отримати бачення про ціле як сукупність частин можливо лише при розгляді архітектури, а не кодуванні. Розглянемо модель проекту, діаграму IDF0, рисунок 3.

Кожна модель має своє призначення. У цьому випадку збирання статистики та надання кращої аналітики автоматизовані. Модель Hospital Manager Helper має чотири сторони, які відповідають за вхід, вихід, контроль та механізм. З лівого боку модель отримує форму даних та запит. Знизу

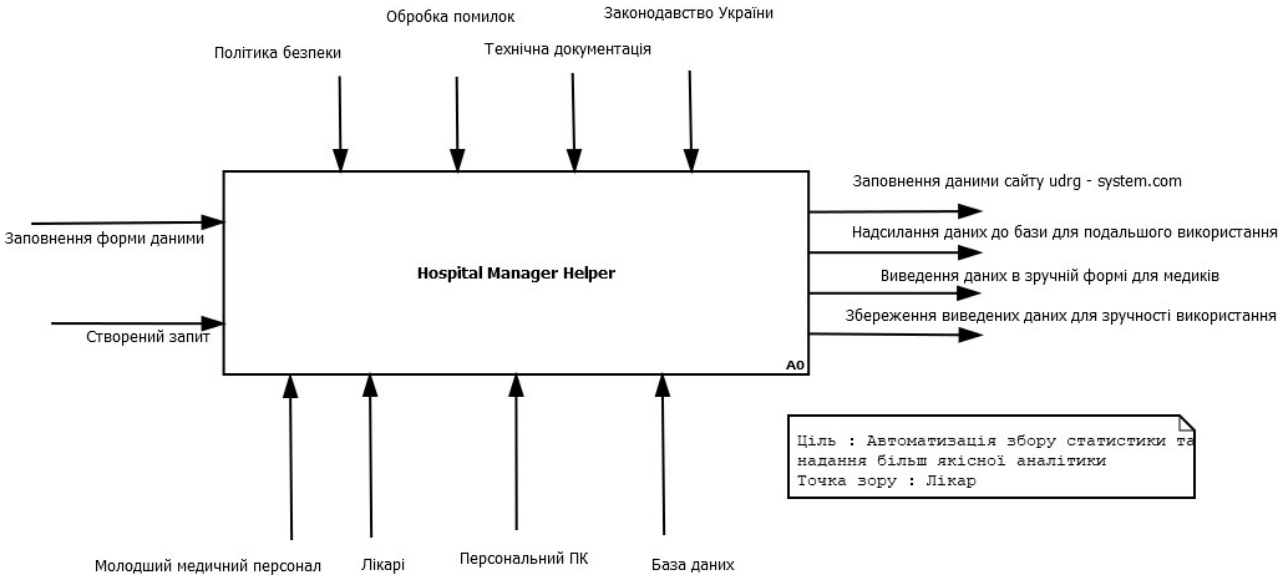


Рисунок 3 – Модель проекту, діаграма IDF0

приєднується механізм, за рахунок чого здійснюється виконання. Механізмом виступає медичний персонал, ПК та база даних. Згори на модель накладається контроль. Всі роботи контролюються вказівками щодо безпеки, поводження з помилками, технічною проектною документацією та законодавством України. З правої сторони моделі виходить автоматично заповнені статистичні форми, зберігання діагностичної інформації в базі, вивід інформації у зручній формі для

подальшого використання та додаткового можливість збереження частини результатів за необхідності.

2.3) Використані технології та засоби

Для створення проекту Hospitel Manager Helper було використано:

- ✓ Мова java;
- ✓ JDBC, який зв’язує за стосунок з базою;
- ✓ IntelliJ IDEA IDE, середовище для розробки програм;
- ✓ GitHub, контроль версій проекту;
- ✓ MS SQL, T-SQL;
- ✓ Maven, для збору проекту;
- ✓ Swing, AWT для реалізації інтерфейсу;
- ✓ jdatepicker – використовується як календар для роботи з датами;
- ✓ selenium – використовується для автоматичного заповнення сайту та тестування проекту;
- ✓ pdfbox – використовується для генерації pdf звітів.

Програмування включає управління складністю, бо складність проблеми перетинається зі складністю машини, на якій вона розрахована. Hospital Manager Helper був написаний об'єктно – орієнтованою мовою Java. Мова програмування Java, створена компанією Sun Microsystems у 1995 році [6].

При створенні мови програмування Java було використано п'ять принципів:

- ➢ бути «простою, об'єктно-орієнтованою та пізнаваною»;
- ➢ бути «надійною та захищеною»;
- ➢ бути «нейтральною для архітектури та портативною»;
- ➢ виконуватись з «високою продуктивністю»;

- бути «інтерпретованою, потоковою та динамічною».

Важливою метою дизайну, який був ключовим фактором раптової популярності Java, є мобільність. У цьому контексті «мобільність» означає, що код, написаний на Java, може бути виконаний на будь – якому апаратному забезпеченні, де використана будь – яка операційна система.

Java була побудована, як виключно об'єктно – орієнтована мова програмування, що дозволяє створити ефективний, організований та потужний код. Простіше кажучи, Java – це багато потокова, об'єктно-орієнтована, незалежна від платформи мова програмування. Це означає, що Java програми можуть виконувати кілька завдань, використовуючи об'єктно – орієнтовані концепції, які можуть працювати на всіх платформах та операційних системах. Це найважливіший фактор, що відрізняє Java від інших мови.

Java допомагає нам розробляти звичайні настільні програми, мобільні додатки та веб-додатки через використання окремих пакетів, таких як пакет J2ME для розробки мобільних додатків і J2EE – пакет для розробки веб-додатків.

Об'єднана простота та потужність Java відрізняє її від інших мов програмування. Java проста тим, що вона не чекає занадто багато від програміста з точки зору управління пам'яттю[7].

Програма Java запускається через віртуальну машину Java (JVM), яка по суті є програмним забезпеченням, реалізація операційної системи, яка використовується для виконання програм Java. Компілювання (процес перетворення коду в читані інструкції для комп'ютера) аналізує код Java та перетворює його в байт-код, який потім дозволяє комп'ютеру зрозуміти інструкції і виконувати їх відповідним чином.

Розподіл платформи Java відбувається у двох пакетах: середовище виконання Java (JRE) і Java Development Kit (JDK)[8]. JRE по суті є віртуальною машиною Java (JVM) запускає програми Java. З іншого боку, JDK - це повнофункціональний

комплект для розробки програмного забезпечення включає JRE, компілятори, інструменти тощо.

Випадковому користувачеві, який хоче лише запускати програми Java на своїй машині, потрібно буде лише встановити JRE[9], оскільки він містить JVM, що дозволяє виконувати програми Java.

На Java є дві бібліотеки графічних компонентів: AWT та Swing. AWT містить усі компоненти, необхідні для проектування графічних інтерфейсів користувача. Однак використання AWT є нелегким, і бібліотека не позбавлена помилок. Компоненти бібліотеки Swing з ними легше працювати і набагато краще реалізуватися. Деякі компоненти Swing потребують класів з бібліотеки AWT. Для їх використання у нас є можливість імпортувати їх.

```
import java.awt.*;
import javax.swing.*;
```

Програми з графічними інтерфейсами користувача (GUI) керуються подіями. Це означає що програма реагує на події. Приклади подій - натискання клавіші, переміщення миші, натискання кнопки та вибір пункту меню.

Програма з графічним інтерфейсом має (коротку) стадію запуску, в якій GUI побудований, але ще не відображається. Деякі інші частини коду (не пов'язані з графікою) також виконуються. Після цього етапу на екрані відображається графічний інтерфейс, і програма контролюється подіями.

Звичайно, ми хочемо, щоб програма реагувала лише на декілька видів подій, не на всі. Програміст повинен вказати ті події, на які програма повинна реагувати і, якою має бути реакція. У Java це робиться шляхом реалізації listener, які чекають на певні події. Одного разу програміст реалізував listener для конкретного типу події, система виконання буде автоматично повідомляти listener, коли відбувається така подія. Тоді listener виконує бажану дію. Події обробляються в порядку їх виникнення.

У Java існує багато типів подій, такі як кнопки, меню або миша. Події містять інформацію про що сталося, напр. що ініціювало подію (кнопка, меню) і де вона сталося (координати миші). Потім ці відомості використовуються listener.

JDBC – один із найстаріших API програмування на платформі Java. Насправді це перший API корпорації, який з часом став платформою J2EE. Її метою є створення спільної публічної мови для доступу Java до будь-якого двигуна бази даних.

JDK не містить інтегрованого середовища розробки Java (IDE). Тому програміст, який використовує лише JDK, також повинен використовувати текстовий редактор і компілювати свої програми з командного рядка.

IntelliJ IDEA IDE був використаний для написання проекту [10]. Він простий у використанні та значно прискорює розробку продукту. IntelliJ IDEA - це інтегроване середовище розвитку для різних мов програмування (Java, Python, Scala, PHP тощо) від JetBrains. Перша версія програми з'явилася в 2001 році.

IntelliJ IDEA доступний у двох версіях: Ultimate Edition та Community Edition. Перша версія платна, друга повністю безкоштовна. Для розробки Hospital Manager Helper використано IntelliJ IDEA Community Edition. Програма містить редактор, середовище компіляції та виконання для розробки додатків.

GitHub використовувався для синхронізації роботи над проектом [11]. Це технологія управління різними версіями проекту на різних фазах його реалізації.

На сторінці GitHub відображається вміст локального сховища Git. Крім структури дерева файлів, GitHub пропонує деякі додаткові функції, які взяті з Git. GitHub також пропонує трекер версій для кожного сховища.

GitHub також пропонує багато інших функцій, які пропонують більше, ніж просто трекер викликів. Мітки допоможуть вам краще переглядати та класифікувати всі звернення. Завдяки своїй природі, GitHub містить багато метаданих: коміти, які зростають з часом і можуть мати різні версії програмного забезпечення, оскільки команда може обговорювати процес реалізації. Надалі

GitHub запропонує можливість зведення гілок до однієї: завершеного проекту. GitHub може бути інтегрований у будь-який сучасний IDE Intellij або Eclipse.

Мова SQL - одна з найважливіших базових технологій у комп'ютері. Вона підтримує сотні баз даних, включаючи MS SQL. Офіційний міжнародний стандарт SQL був прийнятий та змінений кілька разів. Кожне основне програмне забезпечення для обробки даних використовує SQL. SQL є ядром легендарних баз даних Microsoft, Oracle та IBM. SQL також широко використовується в базах даних з відкритим кодом, таких як MySQL або Postgres, що сприяє руху Linux та руху з відкритим кодом. Спочатку SQL був скромним дослідницьким проектом IBM, але з часом він став відомим як важлива комп'ютерна технологія та ринковий фактор. SQL працює лише з базами даних певного типу, реляційними базами даних, які є основним способом організації даних.

Ця програма використовує локальну базу даних MS SQL 2019. Microsoft SQL Server використовує версію SQL як мову запиту під назвою Transact-SQL (скорочено T-SQL), реалізацію SQL-92 (стандарт ISO для SQL). Microsoft має драйвер JDBC, який дозволяє додаткам на базі Java підключатися до Microsoft SQL Server. [12]

Selenium WebDriver широко використовується для автоматизації веб-браузерів у поєднанні з різноманітними інструменти завдяки своєму акуратному та чистому об’єктно-орієнтованому дизайну. Це простий API, який може допомогти в автоматизації браузера. Однак набагато більше потрібен при використанні для тестування та побудови тестової основи.

Вам знадобиться інтегроване середовище розробки (IDE) або редактор коду для створення нового проекту Java та додавання Selenium WebDriver та інші залежності, щоб створити рамку тестування.

У світі Java Eclipse - це широко використовуваний IDE, а також IntelliJ IDEA.

Maven використовується для визначення структури проекту, залежностей, побудови та управління тестом.

Ви можете використовувати IntelliJ та Maven для створення тестового оточення Selenium WebDriver.

Ще однією важливою перевагою використання Maven є те, що ви можете отримати весь Selenium бібліотечні файли та їх залежність, налаштовуючи файл pom.xml. Maven автоматично завантажує необхідні файли з сховища під час створення проекту.

Як і інші майстри, розробники програмного забезпечення покладаються на свої інструменти для створення додатків. Інтегровані середовища розробників (IDE), інструменти відстеження помилок, інструменти для створення, фреймворки та засоби налагодження, такі як аналізатори пам'яті, відіграють життєво важливу роль у повсякденному режимі розробка та обслуговування якісного програмного забезпечення.

Apache Maven - це система управління проектами з відкритим кодом, заснована на стандартах, що спрощує створення, тестування, звітування та упаковку проектів.

Maven став однією з найбільш широко використовуваних програм у всьому світі. Давайте розглянемо деякі причини, чому Maven такий популярний

Часто, коли ми починаємо роботу над новим проектом, витрачається значна кількість часу на прийняття рішення про макет проекту та структуру папок, необхідних для зберігання коду та конфігурації. Ці рішення можуть сильно відрізнятися між проектами та командами, що може ускладнити ситуацію для нових розробників.

Maven вирішує вищезазначені проблеми, стандартизуючи структуру папок і організацію проекту. Maven надає рекомендації щодо того, де різні частини проекту, такого як вихідний код, тестовий код та файли конфігурації, повинні міститись.

Наприклад, Maven пропонує, щоб весь вихідний код Java був розміщений у папка src \ main \ java. Це полегшує розуміння та переміщення будь – якого проекту Maven.

Крім того, ці конвенції полегшують перехід до нового IDE та початок його використання. Історично, IDE відрізнялися структурою проекту та назвами папок. З Maven ваші проекти послідовно структуровані і стають IDE незалежними.

Більшість Java-проектів покладаються на інші проекти з відкритим кодом. Можна завантажити ці залежності і відстежувати їх версії під час використання у своєму проекті.

Maven пропонує зручний спосіб оголошення цих залежностей у проекті окремий зовнішній файл pom.xml Потім він автоматично завантажує ці залежності та дозволяє використовувати їх у своєму проекті. Це сильно спрощує управління залежностями проекту. Файл pom.xml також може слугувати інструментом документації, що передає ваші залежності та їх версії.

Maven дотримується архітектури на основі плагінів, що дозволяє легко збільшувати та налаштовувати його функціональність. Ці додатки інкапсулюють багаторазову побудову та логіку завдання. Сьогодні є сотні доступних плагінів Maven, які можна використовувати для виконання завдань, починаючи від складання коду до упаковки для створення проектної документації. Maven також спрощує створення власних плагінів, тим самим дозволяє вам інтегрувати завдання та робочі процеси, які є специфічними.

Maven забезпечує єдиний інтерфейс для збору проектів. Ви можете побудувати проект Maven за допомогою лише декількох команд, ознайомившись із побудовою Maven. Таким чином, ви можете легко зрозуміти, як будувати інші проекти Maven. Це дозволяє розробникам більше зосередитися на розвитку.

Maven забезпечує потужний інтерфейс командного рядка для проведення різних операцій. Всі основні IDE сьогодні забезпечують відмінну підтримку

інструментів Maven. Крім того, Maven сьогодні є повністю інтегрований з продуктами постійної інтеграції, такими як Jenkins, Bamboo та Hudson.

Як ми вже згадували, Maven пропонує стандартний макет каталогів для своїх проектів. Коли настає час створити новий проект Maven, вам потрібно створити кожен каталог вручну, і це може легко стомити вас. Сюди приходять архетипи Maven. Архетипи Maven – це заздалегідь задані шаблони проектів, які можна використовувати для генерації нових проектів. Проекти, створені за допомогою архетипів, містять усі папки та файли потрібні, щоб ви рухались далі.

Висновки до розділу

На мою думку для реалізації проекту обрано актуальні технології, що дозволяють створити сучасний продукт за короткий час. Це актуально оскільки перевірка життєздатності проекту можна перевірити на мінімально працюючому і перевіреному прототипі. Наведенні вище технології дозволяють створити такий прототип за короткий час.

Висвітлено принципи розробки інтерфейсу користувача. Інтерфейс має допомагати користувачу з використанням функціоналу програмного комплексу.

Розроблена модель проекту показує роботу системи вцілому та розглядає її з точки зору лікаря.

Наведено опис роботи з такими технологіями:

- Мова java;
- JDBC, який зв’язує за стосунок з базою;
- IntelliJ IDEA IDE, середовище для розробки програм;
- GitHub, контроль версій проекту;
- MS SQL, T-SQL;
- Maven, для збору проекту.

3) Реалізація програмної системи

3.1) Діаграми класів

Розробка структури класів є складовою частиною технічного завдання будь – якого проекту. Діаграми класів дають розробнику зрозуміти приблизну структуру класів та їх залежність один від одного. Кваліфікований підхід до створення документації дозволяє знайти існуючі та уникнути багатьох помилок в подальшому.

Об'єктно-орієнтований підхід робить крок вперед, пропонуючи програмісту можливість представляти завдання у своєму просторі. Цей підхід має загальний характер і не підпадає під будь – які обмеження чи характер проблеми. Проблеми вирішуються за допомогою об’єктів у межах простору завдання. Це гнучка і потужна абстракція. За допомогою ООП ви можете описати завдання в контексті самої задачі, а не в контексті ПК, на якому працює програма. Однак все ж є зв’язок з ПК. Кожен об’єкт схожий на невеликий ПК: у ньому є стани та операції, які він може виконувати. Ця аналогія пов'язана із зовнішнім світом, нас оточують об’єкти із властивостями та поведінкою.

Створений та перевірений клас - корисний блок коду. Код для багаторазового використання - вражаюча перевага об'єктно-орієнтованих мов. Новий клас може містити будь-яку кількість інших типів об'єктів, необхідних для досягнення необхідної функціональності. Якщо ви створюєте новий клас із існуючих класів, цей метод є композицією.

Приклади композиції наведені на рисунках 4 і 5. Якщо клас MainWindow створює та працює з об'єктами класів CaseInputWinow та CaseViewer, а клас CaseViewer має DownPanel та rowSearchData. Наявність композиції дає можливість

за умови відповідного ідентифікатору доступу працювати з включеними об’єктами. Використовувати їх поля та методи.

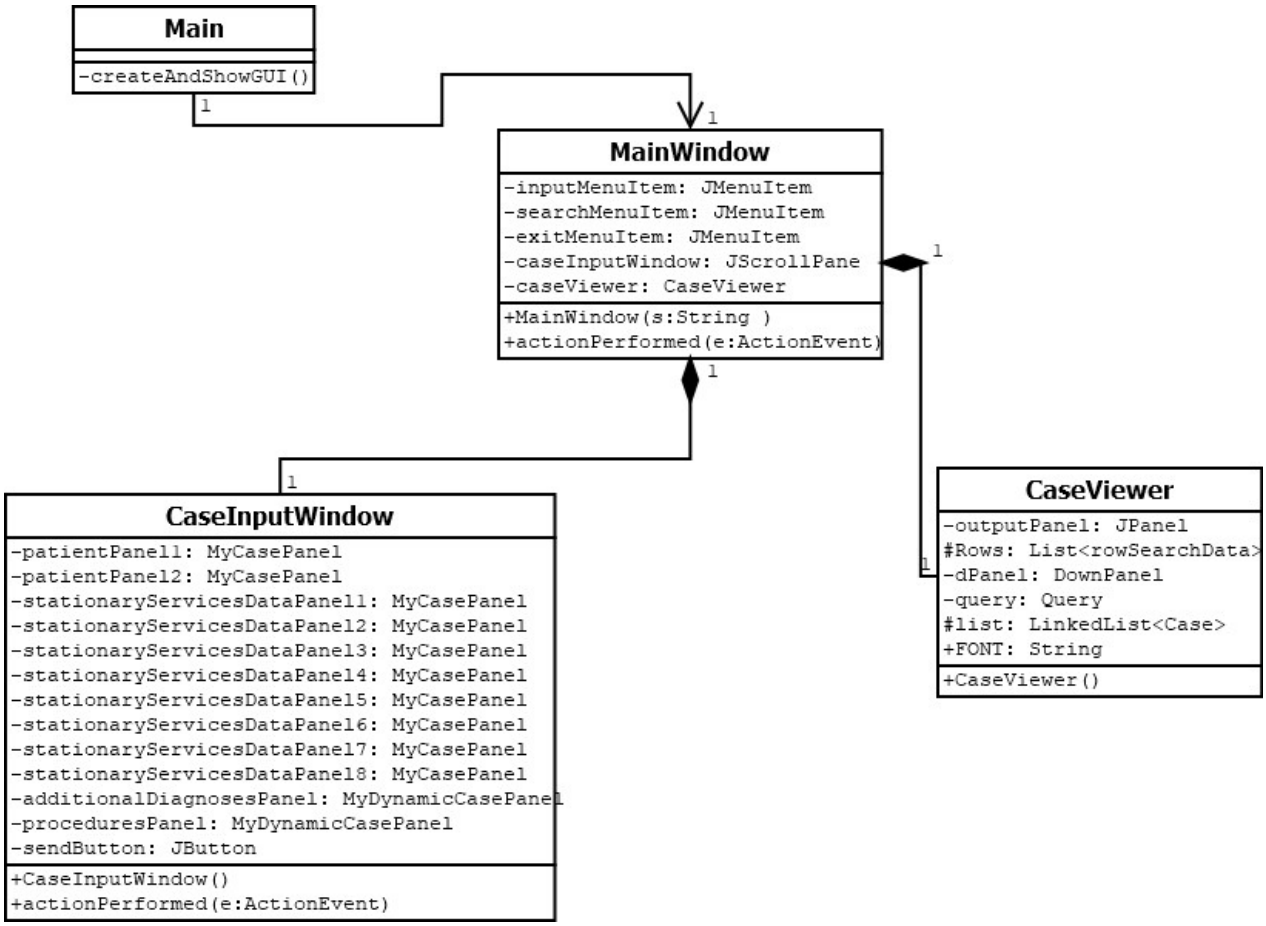


Рисунок 4 – Основні класи

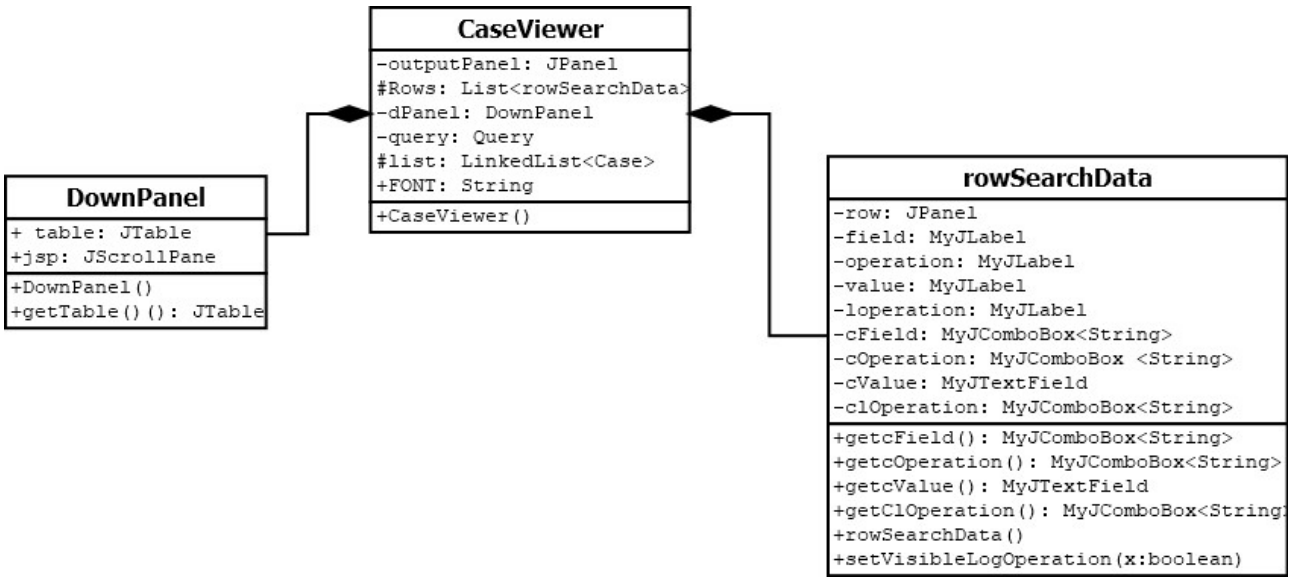


Рисунок 5 – Складові класу CaseViewer

Клас CaseViewer працю є з запитами та відображає їх результати рисунок 6. В процесі роботи з базою важливу участь приймають наступні об’єкти класів: CaseViewer, rowSearchData, Query, DownPanel, SqlApi.

За створення параметрів до запиту відповідає об’єкт класу rowSearchData. Одночасно можна фільтрувати по чотирьох полях бази даних.

Об’єкт класу Query відповідає за створення самого запиту відповідно до параметрів rowSearchData. Цікавий момент, конструктор класу Query за наявності символів «%» або «_» починає шукати подібні значення полів в базі даних.

SqlApi відповідає за безпосереднє з’єднання з базою даних та повернення масиву з результатами запиту.

За відображення отриманого масиву відповідає об’єкт класу DownPanel. Він має в своєму складі таблицю з певним дизайном. Заповнює її в ітеративному режимі проходячи циклом весь масив даних який надійшов з бази даних.

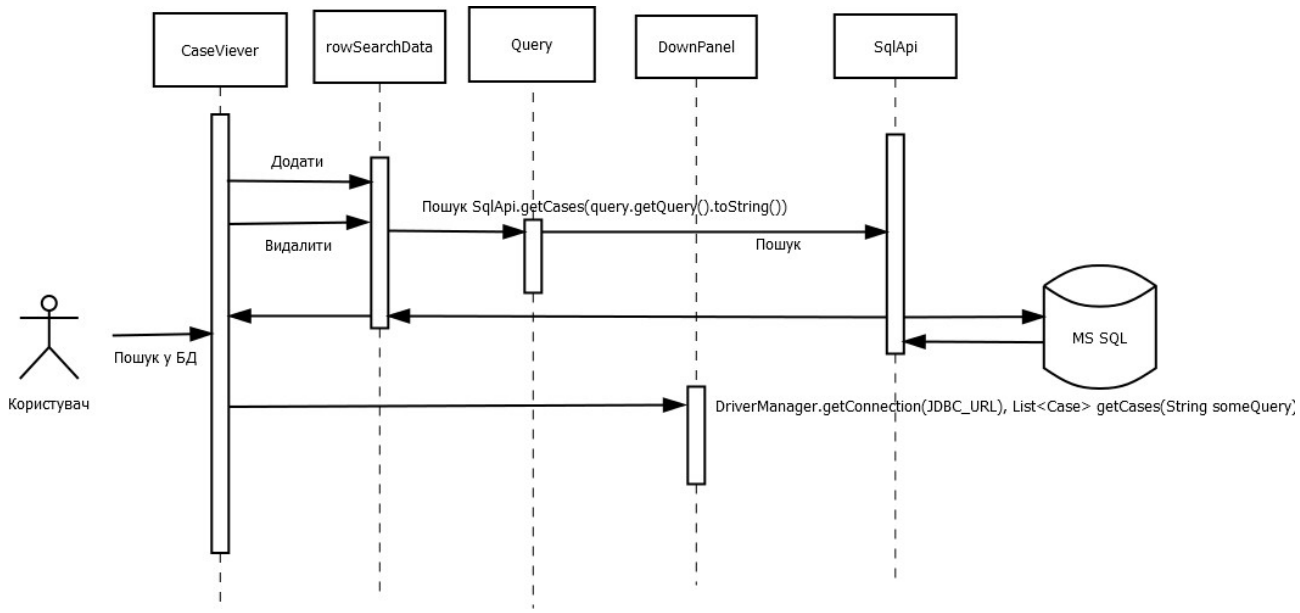


Рисунок 6 – Діаграма послідовності процесу взаємодії з БД

Інкапсуляція – один об’єкт змінює значення змінних іншого об’єкта за допомогою його ж методів. Цей принцип дозволяє зменшити ризик зміни важливих змінних. Так наприклад, клас Case має в своїй структурі get та set методи до кожного поля.

3.2) Розробка програмного продукту

Програмне забезпечення реалізовано мовою Java в IDE ID Intellij IDEA з використанням бібліотек, за допомогою яких можна розширити функціонал. З Maven до проекту, використовуючи залежності, були додані додаткові бібліотеки.

База даних працює в локальній мережі та недоступна для інших мереж. Це зменшує вразливість системи до зловмисників і знижує ризик втрати конфіденційних даних. Однак отримати доступ до сервера можна безпосередньо. Однак у цьому випадку SQL Server налаштований так, що лише один користувач з правами адміністратора має повний доступ до всіх прав. Також є оператор користувача, який дозволяє лише вводити та переглядати дані. Для обох типів потрібні логін та пароль, що значно знижує ймовірність втрати інформації про пацієнта.

Доступ до Інтернету необхідний лише для клієнтської програми та лише з однією метою: подача даних статистики про пацієнта. У випадку відсутності мережі діагностичні дані можна занести в локальну базу. Це дозволить пізніше заповнити веб-форму даними, які вже були збережені в базі даних.

Пошук інформації за допомогою графічного інтерфейсу зручний і простий. Користувач вибирає поле таблиці, для якої хоче отримати базу даних, а потім вибирає операцію над цим полем, значення самого поля. Маю додати, що є можливість додавати дані за допомогою логічних операцій (АБО , І) та робити громіздкий запит до 4 полів включно. Всі елементи інтерфейсу мають підказки які допомагають користувачу обрати необхідну дію та інформацію. Кожне додаткове поле додається або видаляється за допомогою кнопок у графічному інтерфейсі користувача. Додатковою перевагою пошуку є можливість пошуку відповідних даних за допомогою маски (% - багато символів, _ - невизначені символи).

Діаграма прецедентів зображена на рисунку 7.

Запит перетворюється в окремий об’єкт класу Query і надсилається до реляційної бази даних. Сформована відповідь передається до Array List[],з якого заповнюється таблиця та формується звіт PDF.

Array List[] у реалізованому класі є статичним і оновлюється щоразу, коли робиться новий запит. Масив зберігає відповідні Case, які співпали з запитом. Кожний Case містить усі поля про зареєстрований випадок звернення пацієнта.

Він також реалізований для збереження звіту у форматі PDF. Звіт містить інформацію, ідентичну таблиці програми.

Клас SqlApi відповідає за з'єднання з базою даних і має відповідні поля. Він використовує методи для створення запитів та отримання відповідей із бази даних. Клас використовує об'єкти, такі як Query та Case для підтримки обміну інформацією.

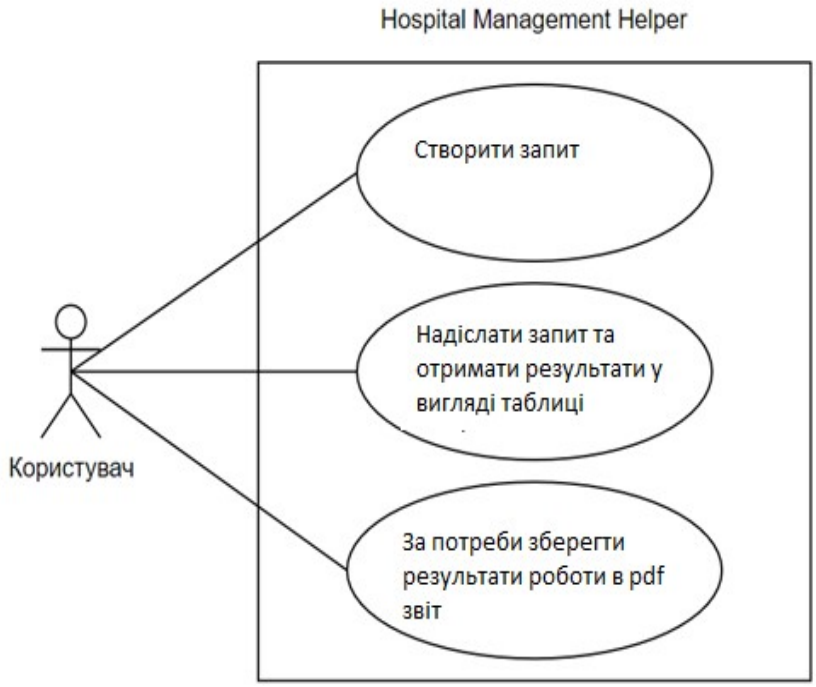


Рисунок 7 - Діаграма прецедентів

Розроблена програма налічує наступні класи рисунок 8.

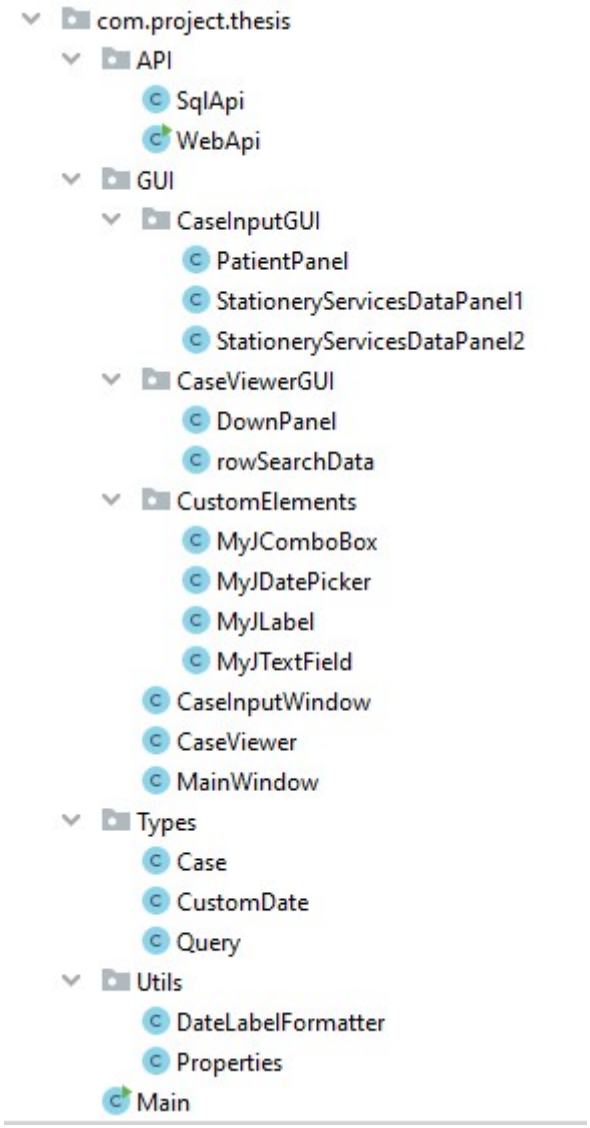


Рисунок 8 - Структура класів

Щоб графічні елементи Swing виглядали краще на основі стандартних елементів з попередньої графічної бібліотеки, ми створюємо власні графічні елементи, які будуть використані в проекті. Ці графічні об'єкти описані в наступних класах: MyComboBox, MyJDatePicker, MyJLabel, MyJTextField.

Класи CaseInputWindow, PatientPanel, StationeryServicesDataPanel1, StationeryServicesDataPanel2 відповідають за заповнення форми програми, сайту та додавання даних про пацієнтів до бази даних.

Класи CaseViewer (додаток А), rowSearchData та DownPanel відповідають за відображення інформації та створення запитів. DownPanel містить елементи з бібліотеки Swing і відповідає за графічне представлення даних, отриманих у відповідь на запит бази даних у табличній формі. У таблиці містяться заголовки, які відповідають усім полям таблиці бази даних. Локальна база даних опитується за допомогою класу rowSearchData, який містить поля, що беруть участь у створених запитах до бази даних. Форма фільтра може містити до чотирьох об'єктів цього класу, які з'єднані логічним АБО або І. Клас CaseViewer містить пошукові лінії, кнопки зміни їх кількості, панель управління з кнопками пошуку та збереження та об’єкт DownPanel, який відображає таблицю.

Існує два основні класи, Case та Query для роботи з даними. Випадок містить поля, які відповідають полям у таблиці бази даних, з якою працює програма. Він також включає методи, які витягують і задають значення кожного окремого поля в класі, а також метод, який повертає об'єкт Case у правильну форму для надсилання в запиті. Клас Query містить усі поля рядків пошуку, методи складання окремих значень поля та методи, які можна використовувати для виконання гнучких запитів, використання масок та створення складних запитів.

Кожний випадок має декілька дат. Заповнення дати реалізовано у вигляді календаря. У цьому нам допомагають класи DateLabelFormatter та CustomDate.

Проект містить багато кнопок з вибором певних значень (JCombobox). Ці значення задаються в масивах і містяться в класі Properties. Оскільки більша частина інформації, яку користувач може ввести у програмну форму, відома заздалегідь, швидкість заповнення форми зростає.

Клас MainWindow додає меню програми до проекту та обробляє дії користувача з будь-яким значенням меню.

Main клас відповідає за створення головного вікна проекту MainWindow та коригування шрифтів та розмірів вікна.

Клас WebApi створений для заповнення форм на сайті та заповнення даних. Для автоматичного заповнення статистичної форми на сайті використовується бібліотека Selenium. Користувачеві потрібно лише підтвердити правильність заповнення сайту даними.

Висновки до розділу

За допомогою вище наведених засобів та бібліотек було запрограмовано Hospital Manager Helper. Структура класів відповідає технічній документації. Реалізовано принципи об’єктно – орієнтованого програмування в проекті.

Проект вцілілому відповідає технічним вимогам та виконує наступні завдання:

- створення модулю авторизації та головного меню;
- введення статистичних діагностичних даних;
- збереження в локальній базі;
- автоматичне заповнення статистичної форми сайту;
- створення запиту до бази даних, можливість створення складних конструкцій;
- форматування отриманих даних від бази даних та заповнення у таблицю;
- обробка помилок вивід повідомлень;
- виведення підказок;
- можливість збереження частини результатів окремо від бази даних.

4) Інструкція користувача Hospital Manager Helper

Перед встановленням програмного забезпечення на ПК потрібно провести роботи з підготовки. Пересвідчитись що виконані наступні вимоги:

- операційна система *Windows 7, 8, 10*;
- *2* гб оперативної пам’яті;
- *200* гб вільного місця на жорсткому диску;
- встановлений java runtime environment 8 або вище;
- встановлений та налаштований MS SQL Server;
- наявна папка за розташуванням C:\temp, для зберігання pdf файлів.

У встановленій та налаштованій базі даних має міститися таблиця Operations з відповідними полями та типами, рисунок 9.

Column Name	Data Type	Allow Null
ID	int	☐
NAME	varchar(MAX)	☑
LAST_NAME	varchar(MAX)	☑
BIRTHDAY	date	☐
GENDER	varchar(25)	☐
INSURANCE_NUMBER	varchar(MAX)	☑
COUNTRY	varchar(MAX)	☑
PASSPORT_DATA	varchar(MAX)	☑
MEDICAL_HISTORY_ID	varchar(MAX)	☐
HOSPITALIZATION_DATE	date	☐
VOLUNTARY_HOSPITALI...	varchar(MAX)	☐
SINGLE_DAY_STATUS	varchar(MAX)	☐
HOSPITALIZATION_TYPE	varchar(MAX)	☑
ADMISSION_REASON	varchar(MAX)	☐
SAPS_TWO_SCORE	varchar(MAX)	☑
HOSPITAL	varchar(MAX)	☐
NHS_OFFICE_CODE	varchar(MAX)	☑
DOCTOR_NAME	varchar(MAX)	☑
REFERRING_DOCTOR_NA...	varchar(MAX)	☑
PRINCIPAL_DIAGNOSIS	varchar(MAX)	☐
TYPE_OF_CARE	varchar(MAX)	☑
HOURS_OF_MECHANIC...	varchar(MAX)	☑
NEWBORN_WEIGHT	varchar(MAX)	☑
DISCHARGE_DATE	date	☐
DISCHARGE_MODE	varchar(MAX)	☐
DISCHARGE_DEPARTMENT	varchar(MAX)	☑
SAPS_TWO_SCORE_AT_D...	varchar(MAX)	☑
TRANSFER_TO_HOSPITAL	varchar(MAX)	☑
ADDITIONAL_DIAGNOSES	varchar(MAX)	☑
PROCEDURES	varchar(MAX)	☑
CREATED_AT	datetime	☐
SENT	varchar(MAX)	☐

Рисунок 9 – Структура таблиці Operations

Після запуску програми Hospital manager Helper необхідною умовою для подальшої роботи є проходження авторизації, рисунок 10. Це дає можливість наділити користувача відповідними правами доступу. Система містить операторів та адміністратора, що має повний доступ до бази та має можливість видаляти та додавати операторів.

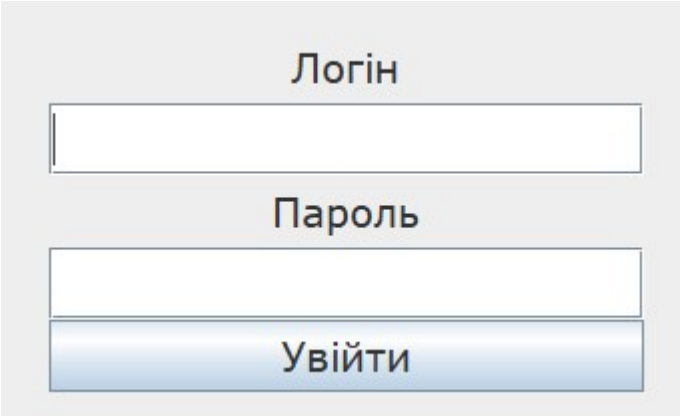


Рисунок 10 – Форма авторизації

Після авторизації, натисніть на кнопку «Меню» у лівій верхній частині екрану та оберіть «Введення даних про випадок», рисунок 11. Це дасть можливість почати вводити дані про випадок. Відкриється форма, обов’язкові поля будуть відмічені зірочкою (*), після заповнення потрібно натиснути

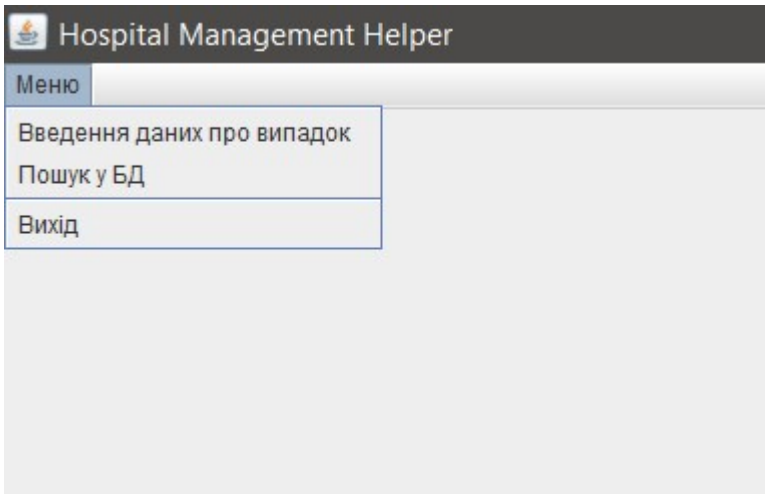


Рисунок 11 - Меню програми

«Відправити», рисунок 12. Після заповнення форми дані надсилаються до локальної бази даних встановленої та налаштованої раніше та сайту збору статистики. Відкриється браузер, який самостійно заповнить форму на сайті https://udrg-system.com та повідомить користувача про необхідність перегляду введеного та остаточне підтвердження відправки.

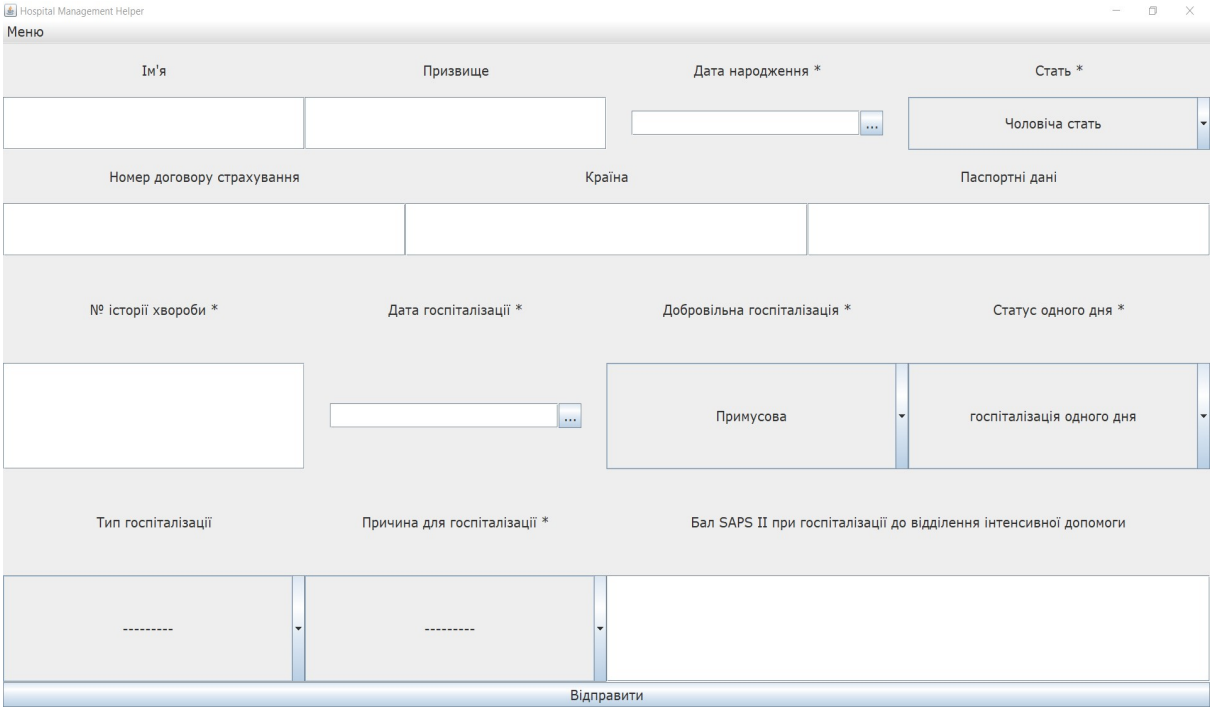


Рисунок 12 – Відправка форми

Якщо користувач прагне переглянути раніше введені дані та провести аналіз. Йому потрібно, після запуску програми, натиснути на кнопку «Меню» у лівій верхній частині екрану та обрати «Пошук у БД», рисунок 11.

Користувач потрапить на сторінку, яка буде містити рядок для створення запиту, з можливістю розширення їх кількості до чотирьох, рисунки 13,14,15

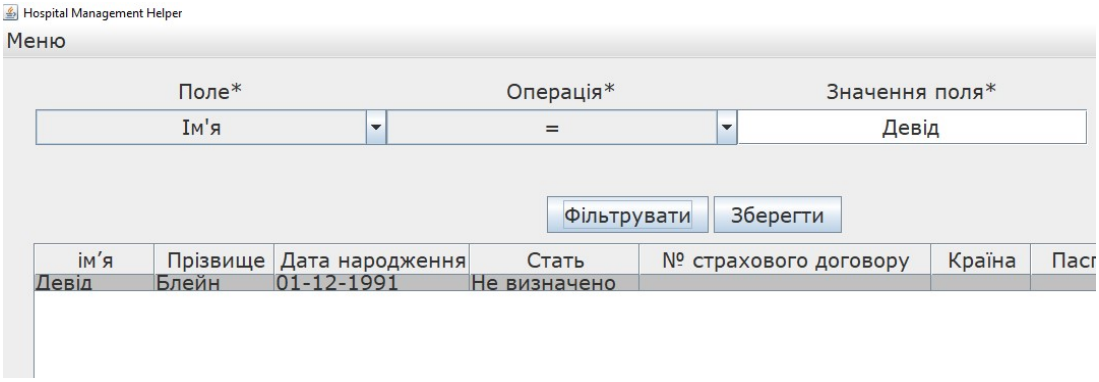


Рисунок 13 - Фільтр та зберігання даних до звіту

Рядки об'єднуються логічними операціями кон'юнкції «І» та диз'юнкції «АБО». Також при пошуку використовується заперечення – інверсія.

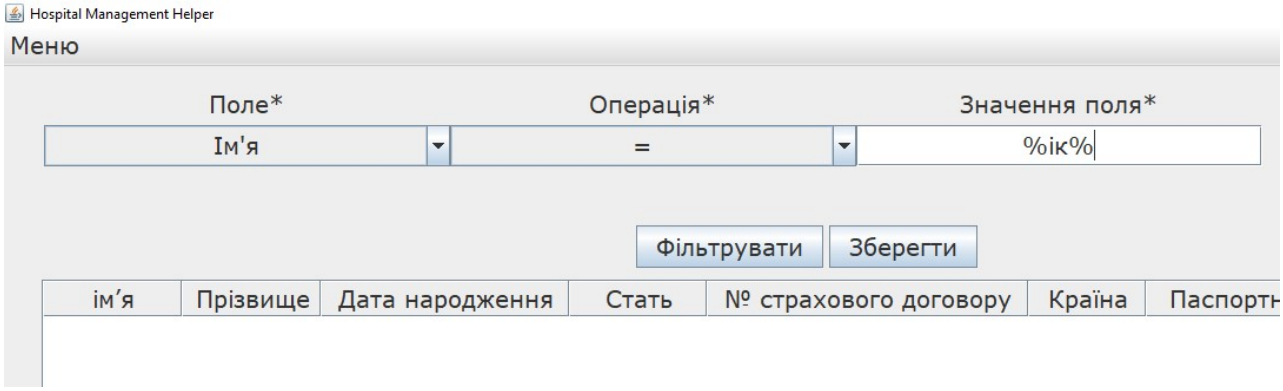


Рисунок 14 - Пошук за маскою

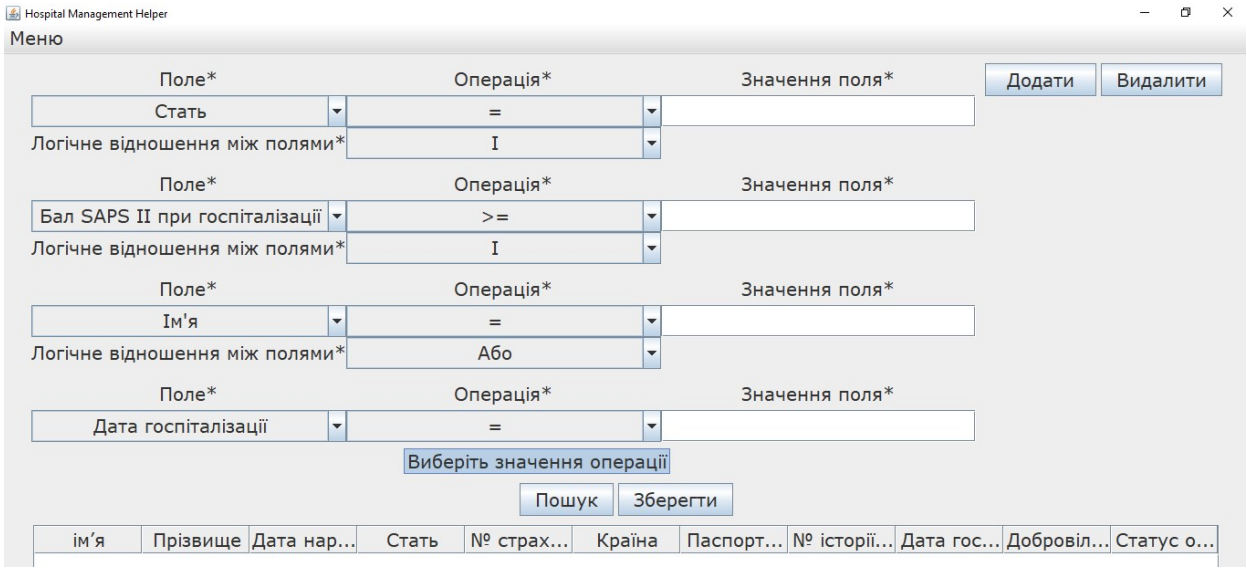
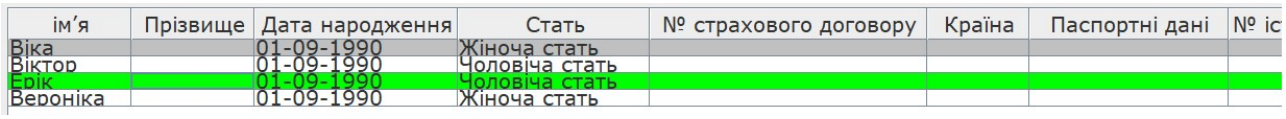


Рисунок 15 – Пошук, 4 поля

Аби збільшити гнучкість пошуку програма фільтрує символи «%» та «_» та дозволяє створювати запити без точної інформації про пацієнта, рисунок 14.

Hospital Manager Helper отримує відповідь на свій запит від локальної бази даних у вигляді масиву. Заповнює таблицю згідно цього масиву, рисунок 16. Для зручності відображення рядки мають різні кольори та для виділення виділяються зеленим кольором. Стовпці таблиці можна міняти місцями разом з вмістом всіх рядків для зручності роботи з даними.



ім'я	Прізвище	Дата народження	Стать	№ страхового договору	Країна	Паспортні дані	№ іс
Віка		01-09-1990	Жіноча стать				
Віктор		01-09-1990	Чоловіча стать				
Ерік		01-09-1990	Чоловіча стать				
Вероніка		01-09-1990	Жіноча стать				

Рисунок 16 – Виведення даних до таблиці

Якщо користувач вирішить зберегти результати роботи він може зберегти їх до pdf файлу зберігаючи структуру таблиці та данні які вона містить. Це дозволяє зберегти частину результату роботи та продовжити її в подальшому.

Програмний комплекс «Hospital Management Helper» допомагає автоматизувати заповнення, захист та збереження даних госпіталізованих хворих лікарні. Система дає можливість аналізу для прийняття рішень та збереження частини результатів для подальшої роботи з ними.

Висновки до розділу

В цьому розділі наведено роботу користувача з програмним комплексом.

Історія вчить, що ти можеш створити якісний та швидкий продукт, але він може з крахом провалитися через те, що користувачі не змогли ним користуватися. Будь – яка програма вирішує конкретну проблему користувача, якщо користувач задоволений результатом то продукт життєздатний.

Програма працює в одній із лікарень міста Києва в тестовому режимі. Це дозволить співпрацювати з медичним персоналом у подальшому вдосконаленні програмного продукту.

Програма має модульну систему завдяки ООП, що дозволяє без великих зусиль розширювати функціонал Hospital Manager Helper. В подальшому можна розробити модуль для глибшого аналізу діагностичних даних, порівняння по шаблону, створити модуль пакетного завантаження звернень пацієнтів абощо.

Висновки

Було досліджено проблеми, які стоять перед системою охорони здоров'я. Джерелами інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет. Створено програмний комплекс Hospital Manager Helper. Результати досліджень представлені на конференції «Сучасні проблеми наукового забезпечення енергетики» 2020.

Подальший розвиток та підвищення продуктивності праці в системі охорони здоров'я України просто не можливо уявити. Для цього потрібно більше коштів. В умовах війни та рецесії в економіці бюджет не може собі цього дозволити. В цьому нам може допомогти система страхової медицини. Початок вже покладено, але ще не всі страхові компанії почали співпрацювати.

Наступним має стати масштабне оновлення медичної інфраструктури. Це стосується: наявності комп'ютерів, швидкий доступ до мережі, якісне програмне забезпечення та медичне обладнання загалом. Це надасть швидкий доступ до діагностичних даних пацієнтів та в майбутньому дозволить робити превентивний аналіз захворювань, що дасть можливість вжити попереджуючі заходи.

Як показує досвід використання інформаційних систем [13] Медичним центром Маунт-Сінай (США), впровадження такої системи зменшило кількість повторних заявок пацієнтів на 56 відсотків, за що вони отримали престижну нагороду Davies Award.

Запропоновано впровадження програмного пакету в одній із київських клінік. Це зменшує залежність від інформації про реєстрацію пацієнтів на веб-сайті UDRG-System.com з огляду на їх працездатність та доступність в Інтернеті. Відтепер медичному персоналу більше не доводиться чекати завершення технічних робіт чи відновлення доступу до Інтернету. Вони можуть заповнити анкети в той час, який їм підходить, та надіслати їх після вирішення технічних проблем. .

Надійність зберігання даних є гарантована також у разі пошкодження веб-сайту UDRG-System.com.

З урахуванням вимог до створення медичних інформаційних систем та особистих побажань старшого медичного персоналу запропонована система, що описана в цьому дипломному проекті. Система має приємний та інтуїтивний інтерфейс без зайвих функцій. Інформаційна система програмного забезпечення «Hospital Management Helper» реалізує функції реєстрації пацієнтів, а також зберігання та передачу даних.

Інформаційна система розроблена мовою програмування Java. В якості бази даних використовувався MS SQL.

Створення цієї системи не тільки полегшило роботу лікарів лікарні з документами та підвищило надійність зберігання даних, а й відкрило нові можливості для покращення надання медичних послуг.

В майбутньому створення загальнодержавної експертної системи дасть можливість запустити на її базі штучний інтелект. Це буде абсолютно новий рівень для системи охорони здоров’я.

Список використаних джерел

1. Закон України від 01.06.2010, № 2297-VI "Про захист персональних даних"

2. Порядок функціонування електронної системи охорони здоров'я, затверджений Постановою КМ України, від 25.04.2018, № 411 "Деякі питання електронної системи охорони здоров'я"

3. Zhou, L.L., Owusu-Marfo, J., Asante Antwi, H. et al. “Assessment of the social influence and facilitating conditions that support nurses adoption of hospital electronic information management systems (HEIMS) in Ghana using the unified theory of acceptance and use of technology [Електронний ресурс] — Режим доступу: https://doi.org/10.1186/s12911-019-0956-z.

4. Формування основних напрямків розвитку інформаційних технологій в охороні здоров’я України на основі світових тенденцій [Електронний ресурс] — Режим доступу: http://www.irbisnbuv.gov.ua/cgibin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Ujtm_2011_9_2_3.p

5. Helsi.me [Електронний ресурс] — Режим доступу: https://helsi.me .

6. Java documentation [Електронний ресурс] — Режим доступу: https://docs.oracle.com/en/java/index.html .

7. Stack-Heap [Електронний ресурс] — Режим доступу: https://www.baeldung.com/java-stack-heap

8. Oracle Java[Електронний ресурс] — Режим доступу: https://www.oracle.com/technetwork/java/index.html

9. Java SE, JDK, JRE, Documentation [Електронний ресурс] — Режим доступу: https://www.oracle.com/technetwork/java/javase/downloads/index.html

10. IntelliJ IDEA [Електронний ресурс] — Режим доступу:https://www.jetbrains.com/idea/download/#section=windows

11. Github [Електронний ресурс] — Режим доступу: https://github.com/

12. sql-server-2019 [Електронний ресурс] — Режим доступу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019

13. Electron medical records system [Електронний ресурс] — Режим доступу: https://www.mountsinai.org/about/newsroom/2012/the-mount-sinai-medical-center-wins-prestigious-2012-davies-award-of-excellence-for-its-electronic-medical-records-emr-system.

14. Эккель Б. Филосовия Java. 4-е полное изд. – СПб.: Питер, 2017 – 1168 с.: ил. – (Серия «Класика computer science»).

15. SQL Docs [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?view=sql-server-ver15.

16. Object-Oriented Design [Електронний ресурс] — Режим доступу: https://www.coursera.org/learn/object-oriented-design.

17. UML [Електронний ресурс] — Режим доступу: https://www.uml.org/

18. Ларман Крэг. Применение UML и шаблонов проектирования. 2-е издание.: Пер. с англ. – М.: Издательський дом «Вильямс», 2004. – 624 с.

19. Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. – СПб.: Питер, 2010. – 464с.

20. Achilleas Pipinellis. GitHub Essentials. – Birmingham.: Packt Publishing , 2015. – 190 .

21. Siriwardena Prabath. Maven Essentials. – Birmingham.: Packt Publishing , 2015. – 365 .

22. Джеймс Р. Гроф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство, 3 – е издание: ООО «И. Д. Вильямс» , 2015 – 960с.

23. Брукс Ф. Мифический человеко – месяц или как создаются программные системы. – СПб: Символ – Плюс , 1999. – 304с.

24. Макконнелл С. Совершенный код. Мастер – класс. – М.: Издательство «Русская редакция», 2010. – 896с.

25. Томас Кормен, Чарльз Лейзерсон. Алгоритмы: построение и анализ, 3 – е изд. – М.: ООО «И. Д. Вильямс», 2013. – 1328с.

26. Charles Humble. The Java Garbage Collection Mini-Book – InfoQ, 2015. – 104.

27. Allen B. Downey and Chris Mayfield. Think Java. – Massachusetts.: Green Tea Press, 2019. – 374.

28. Clifford A. Shaffer. Data Structures and Algorithm Analysis. – Blacksburg.: Virginia Tech, 2013. – 601.

29. The Java Language Specification [Електронний ресурс] — Режим доступу: https://docs.oracle.com/javase/specs/.

30. The Swing Tutorial [Електронний ресурс] — Режим доступу: https://docs.oracle.com/javase/tutorial/uiswing/index.html

31. У 2 т. – К. : 7 КПІ ім. Ігоря Сікорського, 2020. – Т. 2. – 160 с.

Додаток А

```
import com.itextpdf.text.*;
import com.itextpdf.text.Font;
import com.itextpdf.text.pdf.BaseFont;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;
import com.project.thesis.API.SqlApi;
import com.project.thesis.GUI.CaseViewerGUI.DownPanel;
import com.project.thesis.GUI.CaseViewerGUI.rowSearchData;
import com.project.thesis.Types.Case;
import com.project.thesis.Types.Query;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class CaseViewer extends JPanel  {
    JPanel outputPanel = new JPanel();
    List<rowSearchData> Rows = new ArrayList<>();
   DownPanel dPanel;
   Query query;
   protected static LinkedList<Case> list;
   public static final String FONT = "resources/verdana.ttf";

   public CaseViewer() {

        outputPanel = new JPanel();
        outputPanel.setLayout(new BoxLayout(outputPanel, BoxLayout.PAGE_AXIS));

        rowSearchData rSearch = new rowSearchData();
        Rows.add(rSearch);

        JPanel searchData = new JPanel();

        searchData.setLayout(new BoxLayout(searchData, BoxLayout.PAGE_AXIS));
        searchData.add(rSearch);

        JPanel SearchButton = new JPanel();
        JPanel header = new JPanel();

        JButton addSearch= new JButton("Додати");JButton rmSearch= new JButton("Видалити");

        SearchButton.add(addSearch);SearchButton.add(rmSearch);


        header.add(searchData);
        header.add(SearchButton);
        outputPanel.add(header);

        addSearch.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if(Rows.size()<4){
                Rows.add(new rowSearchData());
                Rows.get(Rows.size()-2).setVisibleLogOperation(true);
                searchData.add(Rows.get(Rows.size()-1));

                outputPanel.revalidate();
            }}
        });
        rmSearch.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if(Rows.size()>=2){
                    Rows.get(Rows.size()-2).setVisibleLogOperation(false);
                    searchData.remove(Rows.get(Rows.size()-1));
                    Rows.remove((Rows.size()-1));
```

```
            outputPanel.revalidate();
        }}
    });

JPanel mPanel = new JPanel();
    JButton bt1= new JButton("Фільтрувати");
    JButton bt2= new JButton("Зберегти");


    mPanel.add(bt1, BorderLayout.CENTER);
    mPanel.add(bt2, BorderLayout.CENTER);

    //Search data
    bt1.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            DefaultTableModel model = (DefaultTableModel) dPanel.getTable().getModel();

            while(model.getRowCount()>0)
            {
                model.removeRow(0);
            }

            list = new LinkedList<Case>();

            StringBuffer query = new StringBuffer();

            for(int i=0; i<(Rows.size()); i++)
            {
                //System.out.println("Done");
                query.append(Rows.get(i).toSQLData(Rows.get(i))+" ");
                query.append(Rows.get(i).getcOperation().getSelectedItem()+" ");
                query.append("'"+Rows.get(i).getcValue().getText()+"'");
                //if(i<(Rows.size()-1)){query.append(Rows.get(i).getcOperation()+" ");}
            }

            //--------------------------------
            list = (LinkedList<Case>) SqlApi.getCases(query.toString());

            //--------------------------------
            Object[] row= new Object[8];

            for(int i=0;i<list.size();i++)
            {
                row[0]= list.get(i).getFirstName();
                row[1]= list.get(i).getLastName();
                row[2]= list.get(i).getBirthDay().toWebFormat();
                row[3]= list.get(i).getGender();
                row[4]= list.get(i).getInsuranceNumber();
                row[5]= list.get(i).getCountry();
                row[6]= list.get(i).getPassportData();
               model.addRow(row);


            }
            outputPanel.revalidate();
        }
    });
    //End button Search data
    //Save data
    bt2.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Document document = new Document();
            Font f = FontFactory.getFont(FONT, BaseFont.IDENTITY_H, true);
            Paragraph head = new Paragraph("Список пацієнтів", f);
            head.setAlignment(Element.ALIGN_CENTER);

            PdfPTable pdfTable = new PdfPTable(7);
            pdfTable.setSpacingBefore(20);

            for (int i = 0; i < list.size(); i++) {

                pdfTable.addCell(new Paragraph("" + list.get(i).getFirstName(), f));
                pdfTable.addCell(new Paragraph("" + list.get(i).getLastName(), f));
```

```
                pdfTable.addCell(new Paragraph("" + list.get(i).getBirthDay().toWebFormat(), f));
                pdfTable.addCell(new Paragraph("" + list.get(i).getGender(), f));
                pdfTable.addCell(new Paragraph("" + list.get(i).getInsuranceNumber(), f));
                pdfTable.addCell(new Paragraph("" + list.get(i).getCountry(), f));
                pdfTable.addCell(new Paragraph("" + list.get(i).getPassportData(), f));

            }
            try {
                PdfWriter.getInstance(document, new FileOutputStream("C:\\temp\\zvit.pdf"));
            } catch (DocumentException ex) {
                ex.printStackTrace();
            } catch (FileNotFoundException ex) {
                ex.printStackTrace();
            }
            document.open();
            try {
                document.add(head);
                document.add(pdfTable);
            } catch (DocumentException ex) {
                ex.printStackTrace();
            }
            document.close();
        }
    });
    //end Button save data
  outputPanel.add(mPanel);


  dPanel = new DownPanel();
  outputPanel.add(dPanel);


        add(outputPanel);

    }
}
```

Додаток Б

СИСТЕМА АНАЛІЗУ ТА ФОРМУВАННЯ ЗВІТІВ СТАЦІОНАРНОГО ЛІКУВАННЯ ХВОРИХ

Тези наукової конференції

Аркушів 1

2020 р.

УДК 004.021: 004.422.833
Студент 4 курсу, гр. ТВ-з61 Паньківський О.В.
Доц., к.т.н. Крячок О.С.

СИСТЕМА АНАЛІЗУ ТА ФОРМУВАННЯ ЗВІТІВ СТАЦІОНАРНОГО ЛІКУВАННЯ ХВОРИХ

Організації в галузі охорони здоров'я в усьому світі продовжують інвестувати в інформаційні технології. Документація у звіті про стан здоров'я пацієнта повинна відображати оцінку, діагностику, ефекти, планування, дії та оцінку догляду за хворим. Медичні послуги є тією сферою, на яку суттєво впливає впровадження інформаційних технологій. Перетворення паперових документів у цифрові форматовані медичні записи, які містять загальну інформацію про пацієнта, стан здоров'я та спостереження за хронічними захворюваннями має бути невід'ємною частиною ефективної системи охорони здоров'я [1].

Головне призначення запропонованої системи полягає в тому, щоб записи могли зберігатися протягом більш тривалого періоду часу і ними можна було легко обмінюватися між відділеннями та лікарнями. Також враховано той факт, що успішна реалізація будь-яких IT - проектів в галузі охорони здоров'я критично залежить від сприйняття користувачем зручності самої програмної системи.

Оскільки медичний персонал працює на передовій лінії клінічної системи охорони здоров'я з доступом до важливих даних про пацієнтів у лікарнях, то пропонується на базі аналізу проведених досліджень [2] розробити нову інформаційну систему.

Інформаційна система має вирішити наступні основні завдання:

• забезпечити безпеку персональних даних пацієнтів;

• оптимізувати використання ресурсів постійно зростаючої бази даних;

• забезпечити зручний інтерфейс для кожного авторизованого користувача системи.

Впровадження програмного забезпечення в спеціалізованій клініці дало змогу вирішити ці три основних завдання.

Створення всеосяжної та інтегрованої медичної інформаційної системи є необхідною умовою проведення реформ, оскільки це важливо для інтеграції лікарських ресурсів та підвищення якості надання медичних послуг.

Отже, якість надання інформаційної складової медичних послуг потребує подальшого покращення. В роботі запропоновано двоетапну систему внесення електронних записів для більш функціонального використання отриманих даних. Лікарі зможуть швидко отримати доступ до актуальної інформації під час проведення медичних оглядів. Ефективність та якість застосування запропонованої системи підлягає подальшому дослідженню.

Перелік посилань:

1. Про електронні документи та електронний документообіг: Закон України від 22.05.2003 р. No 851-IV. Дата оновлення: 07.11.2018. URL: https://zakon.rada.gov.ua/laws/show/851-15 (дата звернення 10.03.2020).

2. Zhou, L.L., Owusu-Marfo, J., Asante Antwi, H. et al.«Assessment of the social influence and facilitating conditions that support nurses' adoption of hospital electronic information management systems (HEIMS) in Ghana using the unified theory of acceptance and use of technology (UTAUT) model». BMC Med Inform DecisMak. 2019. No 230. DOI:https://doi.org/10.1186/s12911-019-0956-z.